Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №6

Архітектура комп’ютера

«СИСТЕМА ОБРОБКИ ПЕРЕРИВАНЬ АРХІТЕКТУРІ IA-32 (X86) У REAL ADDRESS MODE»

Виконали:

студенти групи ІТ-01:

Бардін В.Д.
Задніпрянець А.А.
Куркін О.О.

Перевірив:

Бердник Ю. М.

Дата здачі 22.04.2021

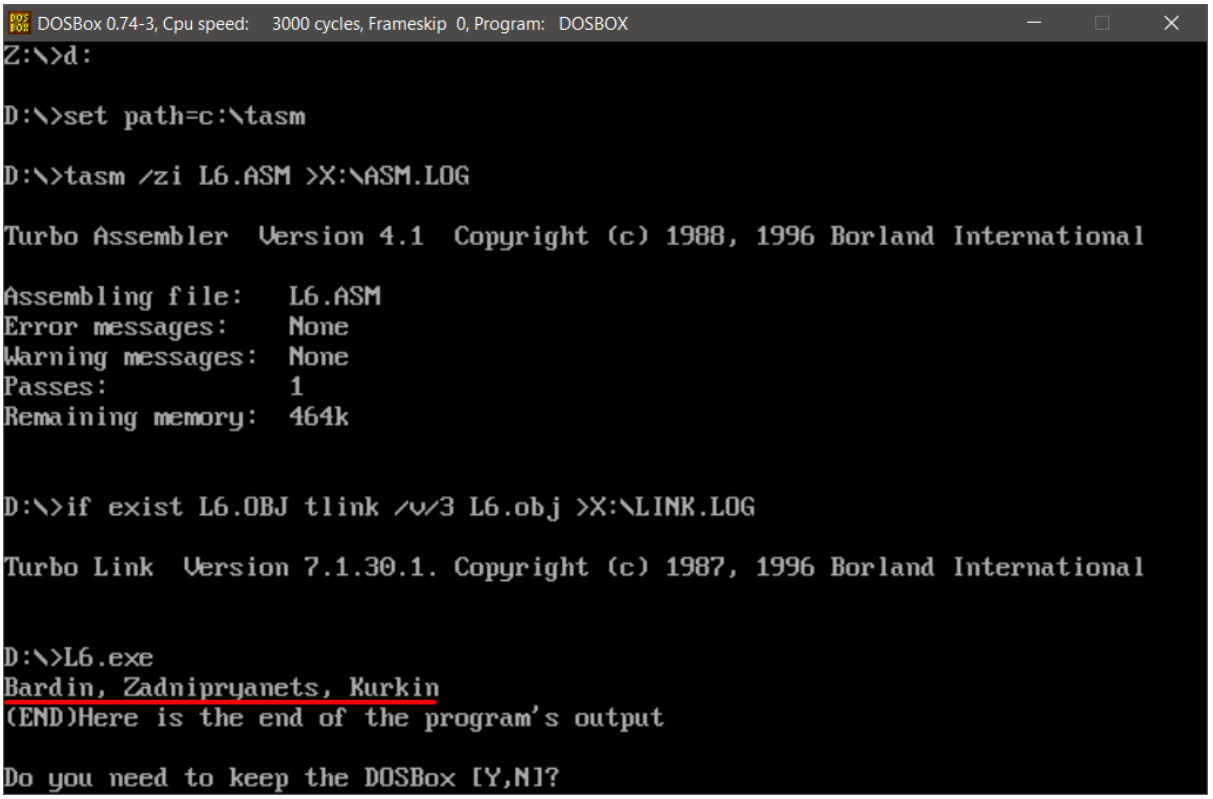
Захищено з балом _____

Київ 2021

Вихідний код:

```
TITLE ЛР_6
;-------------------------------------------------------------------------------
; Дисципліна: Архітектура комп'ютера
; НТУУ "КПІ"
; Факультет: ФІОТ
; Курс: 1
; Група: ІТ-01
;-------------------------------------------------------------------------------
; Автор: Бардін, Задніпрянець, Куркін
; Дата: 07/04/2021
;-------------------------------------------------------------------------------
IDEAL

;-------------------------------------------------------------------------------
MACRO M_Exit ; Exit macros
; Input: AL = Exit code
; Output: ---
mov ah, 04Ch ; DOS interruption number to exit
int 21h
ENDM

;-------------------------------------------------------------------------------
MACRO M_Init ; Initiate DS and ES
mov ax, @data ; ax <- @data
mov ds, ax ; ds <- ax
mov es, ax ; es <- ax
ENDM

MODEL small
STACK 256

DATASEG
msg db "Bardin, Zadnipryanets, Kurkin $"

bak_int0Bh_offset DW ? ; Effective procedure address - default handler for COM1 interrupt
bak_int0Bh_seg DW ? ; Segment begin address for hardware conversion function COM1

CODESEG

PROC main
M_Init

;Stage 1. Receiving effective address and interrupt offset--------------------
mov di, 61h
call get_int_vector
; Output arguments: (bx - effective address, es - segment address )
mov [bak_int0Bh_offset], bx
mov [bak_int0Bh_seg], es

; Stage 2. Збереження стандартного обробника переривань COM 2 за іншим вектором
mov di, 61h                 ; Input parameter for set_int_vector, new interruption number
mov dx, bx                  ; Offset for procedure, effcetive address
call set_int_vector

; Stage 3. Set new handler for custom interruption
mov di, 61h
mov dx, OFFSET int61h
mov ax, SEG int61h
mov es, ax
call set_int_vector

 int 61h                    ; call interruption

; Stage 4. Enroll all changes
mov di, 61h
mov dx, [bak_int0Bh_offset]
mov ax, [bak_int0Bh_seg]
mov es, ax
call set_int_vector

xor al, al                  ; Set al to zero
M_Exit
ENDP main
```

```
;--------------------------------------------------------------------------
PROC get_int_vector
; Purpose: Get logical address for procedure by interruption vector number
; Input: DI <- Interruption vector number
; Вихід: BX <- procedure`s effective address
; ES <- procedure`s segment address
;--------------------------------------------------------------------------
                                ; Add registers to stack
push ax
push di

xor ax, ax                      ; Clean AX
mov es, ax                      ; Move to 0000h
shl di, 2                       ; Left logical shift, same as multiply to 4
mov bx, [es:[di]]               ; Write handler`s effectiva address to BX
mov ax, [es:[di + 2]]           ; Write handler`s segment address to AX
mov es, ax                      ; Write handler`s segment address to ES
                                ; Restore registers
pop di
pop ax
ret                             ; Return from procedure
ENDP get_int_vector

;--------------------------------------------------------------------------
PROC set_int_vector
; Purpose: Set new a handler to interruption number
; Input: DI - interruption number, where the procedure will be bind
; DX - new handler effective address
; ES - new handler segment address
; Output: ---
;--------------------------------------------------------------------------
 Cli                            ; Disable hardware interrupts

                                ; Adding registers to stack
push ax
push di
push ds

xor ax, ax                      ; Clean AX
mov ds, ax                      ; Move to 0000h
shl di, 2                       ; left logical shift, same as multiply to 4
mov [ds:[di]], dx               ; Write effective address to the first half of vector
mov [ds:[di + 2]], es           ; Write segment address to the second half of vector

                                ; Restore registers
pop ds
pop di
pop ax
sti                             ; Enable hardware interrupts
ret
ENDP set_int_vector

;--------------------------------------------------------------------------
PROC int61h
; Purpose: New interruption (61h)
; Input: ---
; Output: ---
;--------------------------------------------------------------------------
 mov ah, 09h                ; print a message
 mov dx, offset msg
 int 21h

 mov al, 20h                ; exit to operation system
 out 20h, al
 iret                       ; return from interruption
ENDP int61h
END main ; Кінець коду
```



Результати дослідження:

Проаналізувавши отриманий код, можна дійти висновку, що переривання це звичайні процедури, які фактично є глобальними й доступні не в контексті однієї програми, а для всіх програм у системі. Також ми створили власне переривання, що реалізує потрібну процедуру.

Висновок: В ході виконання лабораторної роботи, ми розробили власне програмне переривання. Дослідили передумови для створення власного переривання. По-перше, для гнучкості розробки і зменшення обсягу коду. По-друге, для реалізації функції обробки нового апаратного переривання для управління нестандартним обладнанням МПС. По-третє, для вдосконалення або повного переписання процедури обробки переривання, що запрограмована у ОС.